RUTGERS
UNIVERSITY

Independent Study in Computer Science

Implementation of RVO Algorithm

Da Huo

May 9, 2018

1. Introduction

To get rid of the dependency to external libraries, we create our own implementation of the RVO algorithm. In this project we implemented two software, path resampling and the RVO algorithm.

2. Path Resampling

In our previous work, since different path planning algorithms give paths with different densities, our robots cannot run smoothly. In order to let the robots run smoothly while preserving the collosion free path, we need to come up with an algorithm which resamples data points with a fixed density following the original path.

2.1 Algorithm

Given a fixed distance between new path points d, we first calculate the distance between current points in new path to the next point in old path $dist$. We break the old path to n line segments, while $dist > d$, we keep find the equation or slope of current line segment, keep sampling new points on current line segment until $dist < d$, then we move on to the next line segment and repeat this process until we run out of line segments.

2.2 Experiment

We tested our algorithm first on a un-evenly distributed path with a right angle as figure 1 shows:

we get the following results after resampling with $d = 0.5$ as figure 2 shows

We then tested our algorithm first on a un-evenly distributed random path as figure 3 shows:

we get the following results after resampling with $d = 0.5$ as figure 4 shows

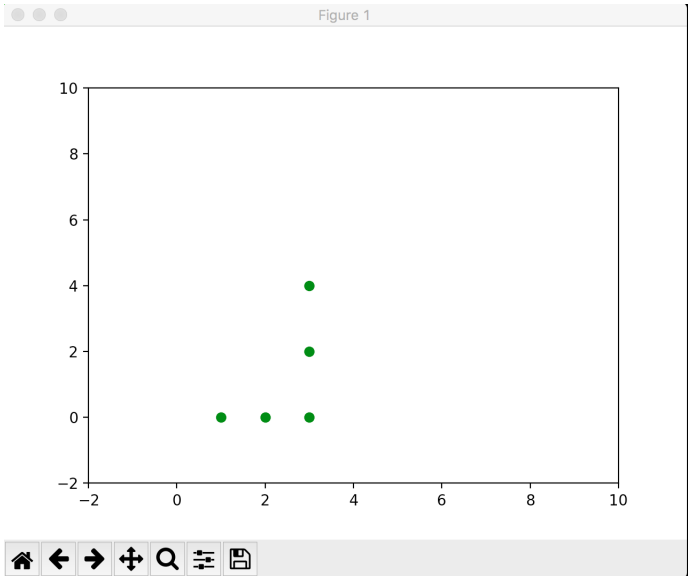

Figure 1: path with a right angle before sample

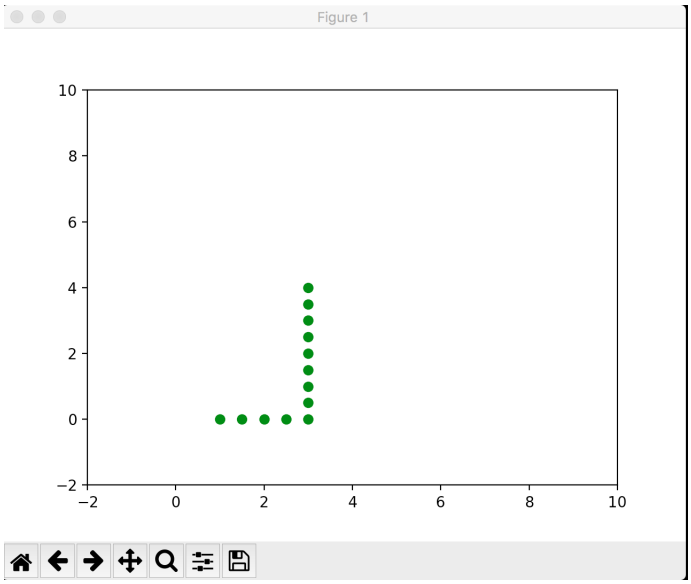

Figure 2: path with a right angle after sample

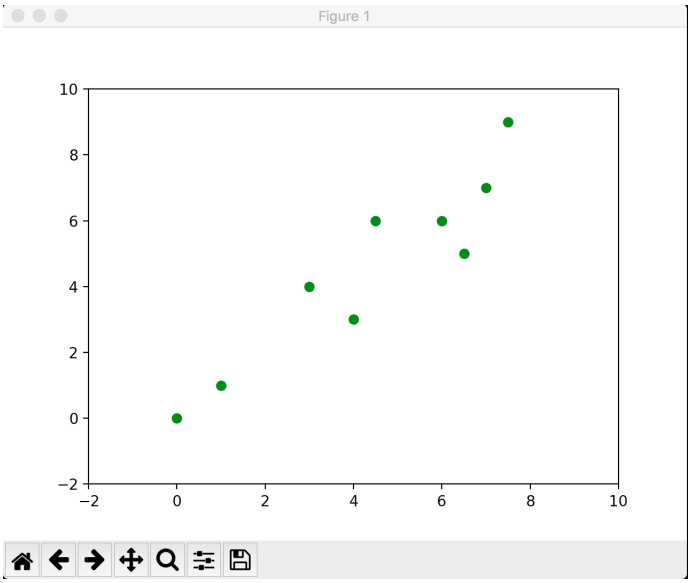

Figure 3: random path before sample

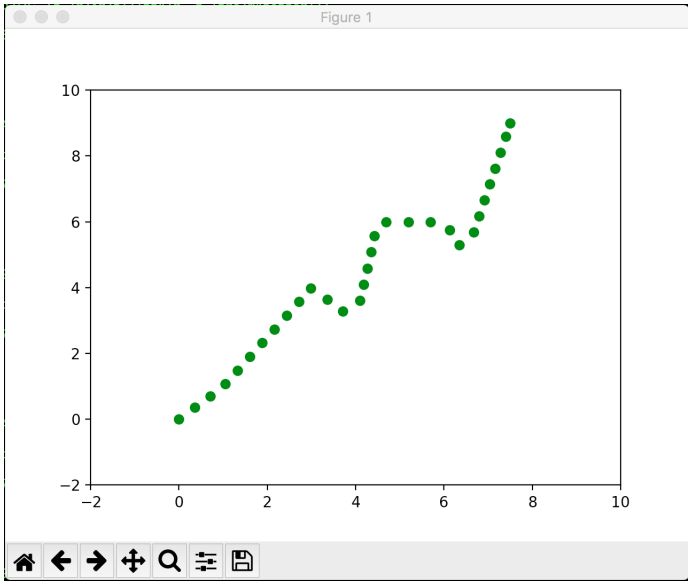

Figure 4: random path after sample

3. RVO

3.1 Algorithm

For every two agents, if the agents will collide within the time horizon t, then we generate a velocity obstacle for each party. After we generate velocity obstacles for each neighbor that are potentially in collision in time t, we find the boundry of the union of these velocities. The orthogonal projection of the agent's prefered velocity onto the cloest point from agent's prefered velocity to the boundry of the velocity obstacles. This projection is the new velocity of our agent.

3.2 Experiment

We tested our program with 2 agents going toward each other as figure 5-8 shows

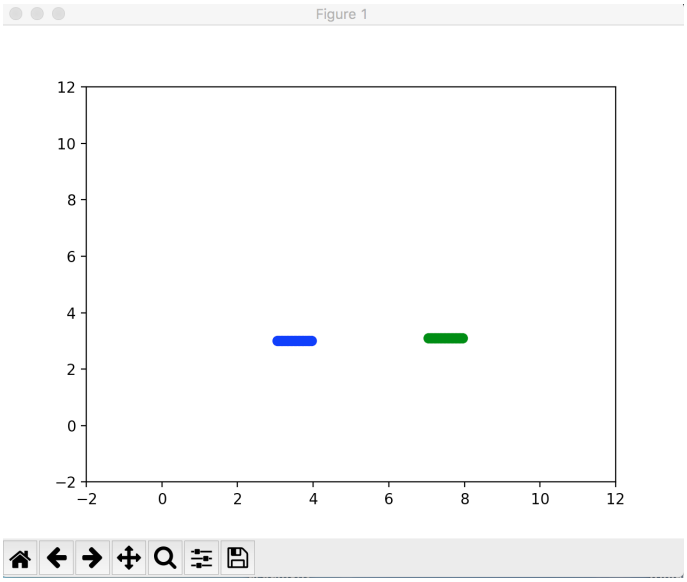

Figure 5: 2 agents going toward each other at time $t1$

We also tested our program with 6 agents going toward each other as figure 9-12 shows

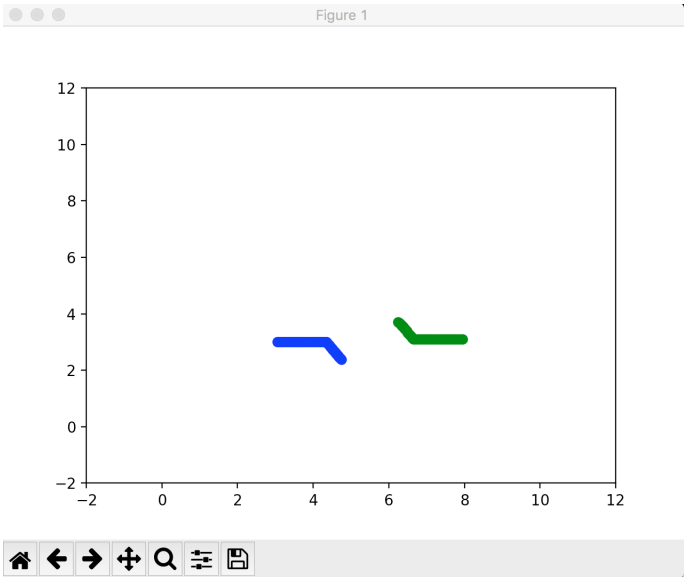

Figure 6: 2 agents going toward each other at time $t2$

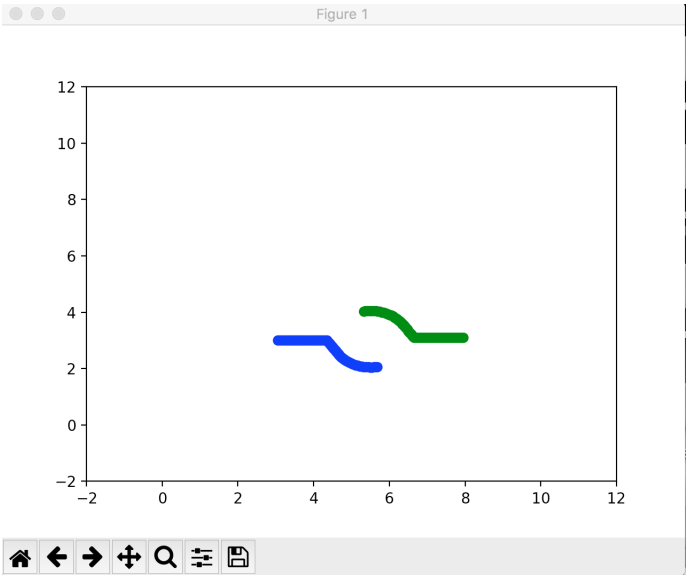

Figure 7: 2 agents going toward each other at time $t3$

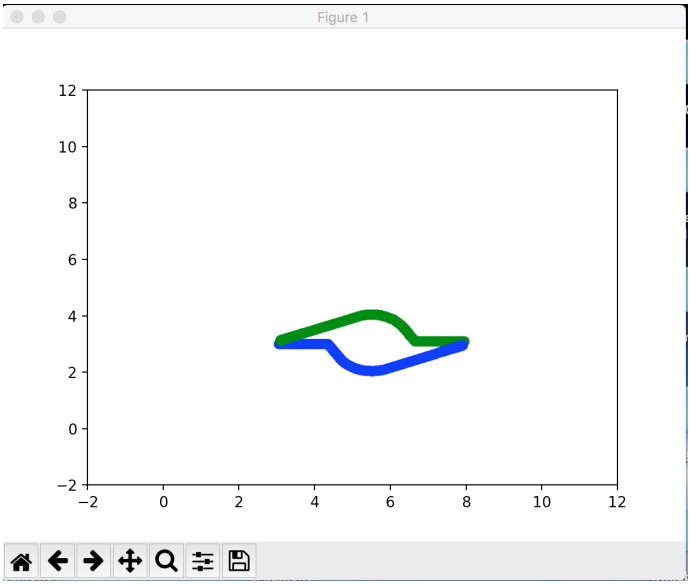

Figure 8: 2 agents going toward each other at time $t4$

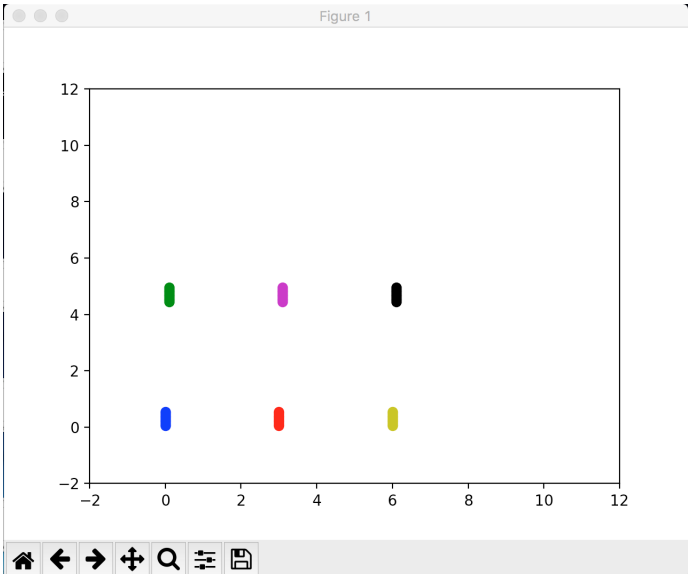

Figure 9: 6 agents going toward each other at time $t1$

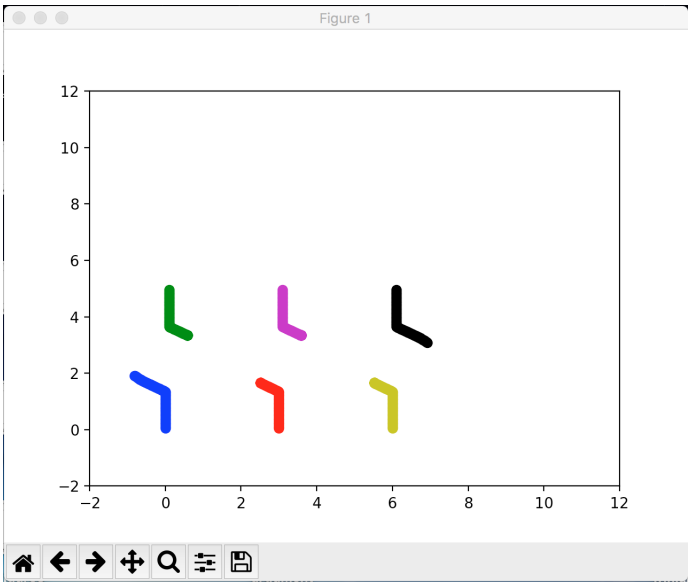

Figure 10: 6 agents going toward each other at time $t2$

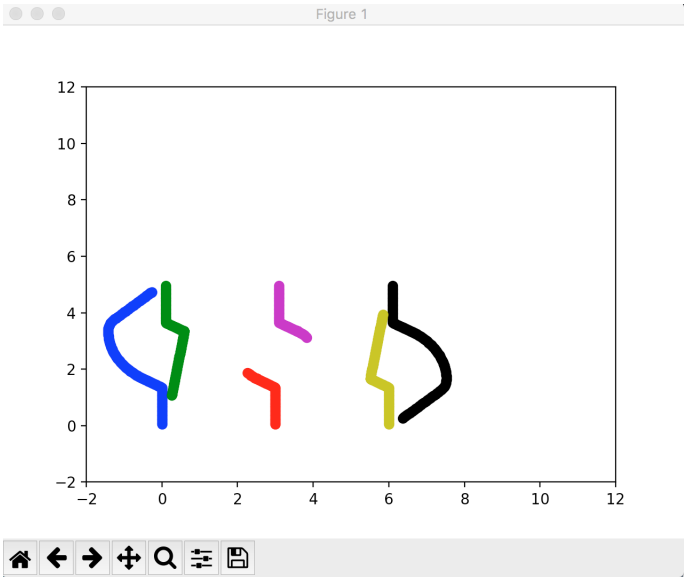

Figure 11: 6 agents going toward each other at time $t3$

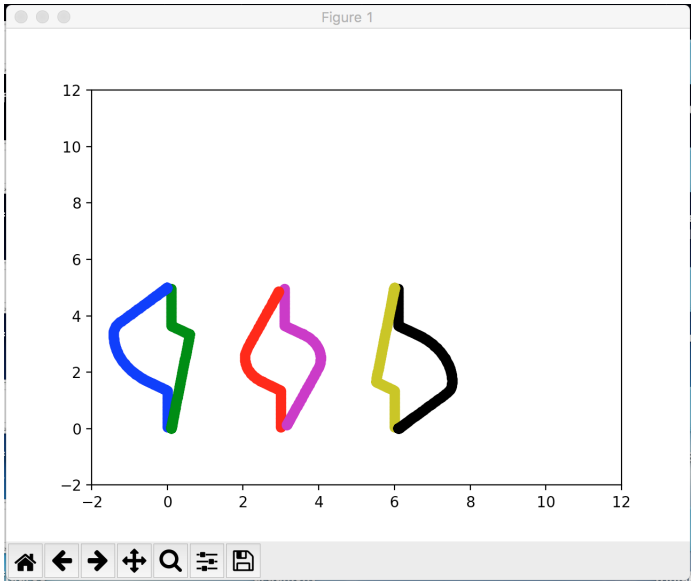

Figure 12: 6 agents going toward each other at time $t4$